CORTEX USERS GROUP

T GRAY,1 Larkspur Drive,Featherstone,Wolverhampton,West Midland WV10 7TN
TEL No 0902 729078

E SERWA,93 Long Knowle Lane,Wednesfield,Wolverhampton,West Midland WV11 1JG
TEL No 0902 732659

CORTEX USER GROUP NEWSLETTER (MARCH 1988)

Issue Number 17

CONTENTS

REMEMBER TO SEND IN YOUR ARTICLES FOR THE NEXT NEWSLETTER

LETTERS

A Rowell Northants comments to newsletter 16

Page 4 Dick Hall., I have disassembled MDEX.REL but I do not have a source copy only a disassembly printout with written comments on. If you need any further information on the internals of MDEX then just ask.
Page 3 O.C.Walden., If the precession table pointer in the device table for a given disk is set to zero all future access to the disk will not use interleaved sectors. This is an easy way to do standard disk accesses. Remember to set it back to its original value after use or it will not be usable by MDEX.

A.C.Heslop Tyne-Wear

My Cortex II has no seperate disk drive case as previously available from Powertran. Can you supply one and if so what is the price and dimentions. Also I am interested in obtaining the MDEX system including Qbasic and Window Ioperate a single 5'' 100K drive is this sufficiant for MDEX. Alternatively does the software exist to run the system with 3.5'' drives.

The nearest boxes we can find are R.S.components stock number 507-731 which is 342 X 260 X 70 at £27.62 or 507-719 at £34.19 which is 430 X 260 X 100 or possibly their 501-626 disk drive case designed to hold two 5'' disk drives mounted one above the other at £24.08 all dimensions are W X H X D and prices exclude V.A.T. Mdex really needs two disk drives to work sensibly and a minimum of 360K per drive. We can't as yet supply MDEX on 3.5'' disks although there is no reason it should not run as long as the drives work in the same way as 5'' or 8'' drives. Note MDEX is only available for the TMS9909 disk controller.

Mike Gallagher Rugby

Can anyone help with my problem. When switching on my Cortex the usual Cortex Basic message comes up as normal, but then after a couple of minutes indiscriminate symbols start to appear all over the screen, and most of them will not even clear when pressing the clear button. The problem seems to be getting worse as time goes on and at the moment my Cortex is unusable. I'v tried removing all the inputs, including the Keyboard, to no avail I'm begining to suspect the CPU. Has anyone any other ideas ?.

We think it highly unlikely that the CPU is faulty but would suspect either the VDP chip or one of its RAM chips or possibly the associated XTAL. Try removing one of the VDP RAM chips at a time you should get a screen full of the same pattern in every location after a clear. If you can achieve this with one RAM chip missing without random symbols apearing then try a new RAM chip.

REMEMBER TO SEND IN YOUR ARTICLES FOR THE NEXT NEWSLETTER

Documentary support for Mdex Pascal is very sparse, or horrendously expensive. Brinch Hansen's book [ref.1.], cost me almost $60, but it is not very helpful. These notes are intended to help in getting started. They should be read alongside the User Manual and a print-out of the Prefix from the master disc. The program Copy included works(!) but was written more to illustrate points raised than to have any enduring usefulness.

The Pascal Program.

This must take the form:-
Prefix, followed by Definitions & Declarations of
Constants, Data Types, Variables and Routines, (Procedures & Functions).
These are followed by the sequence of Statements forming the main program which will be executed one at a time.

The program is automatically stored on disc after compilation, and is activated by typing its name and any parameters on the console. To be pedantic,it is the name of the file containing the program which is typed. It is permitted, but surely confusing, to use another for the program.

The program Copy would be called:-

```
Copy (This_file,That_file)
```

File names must comply with Pascal identifier syntax, but may be standard Mdex names. The drive number on disc files is not taken as part of the Pascal identifier. Device files are named as per Mdex, i.e. Cons.dev, Print.dev, etc. This causes some restrictions, see later.

Each parameter in the list, more correctly called the Argument list, may be referenced by number. The system always inserts an extra param of type boolean which may be used to vet program performance. This extra one is Param[1] (or Param(.1.)); so This_file is Param[2], and so on. Ten arguments are allowed, any not mentioned are set to Niltype. See the record type Argtype in Prefix.

The Prefix.

Used by the authors of the language to hive off all hardware related facilities to ease transportability between different implementations. It consists of dummy routines which define the syntax of calls upon them, and direct execution to the actual routines within the operating system. Some routines must be called before others become effective.

File manipulation.

There are two distinct groups for this, they serve to tell the system the names and intended use of the files.

1. READ/WRITE.
   A call on the Prefix procedure Writearg activates this group:-
   Writearg(inp,source) or Writearg(out, dest) opens
   the files named in the variables Source and Dest
   which must be defined as Argtype.
   This activates the procedures Read, Write, Readpage
   and Writepage which may then be used to access
   their files sequentially from top to eof.
   The closure call is Readarg(inp, source) or Readarg(out, dest).
   They return a boolean to inform on satisfactory
   file usage.
2. Open(fileno,filename,found) associates an integer
   with the file by which it is thereafter referenced.
   This can only take the values 1 or 2, so only
   two files can be open at a time.
   This enables Get, Put and Length. Get & Put can
   set the file pointer to any page so give random access.
   Close terminates access and allows re-allocation of the file number.

Console I/O.

If the console is described to the system as Cons.dev, all the above file routines are applicable. The random nature of Get/Put is ignored. However, the console may be addressed by calls to two special routines, Display and Accept. These transfer single Ascii chars and need no prior activation. (see Procedure Context in Copy).

I/O Ascii restriction.

Pascal can only accept ascii chars so all other types will need conversion. Pasedit on the master disc provides much of what is needed for the console. If its small demo program is deleted, it can be copied to a user program in place of Prefix since that is alrady present. User's global Constants, vars,etc., should be placed between the Prefix and Pasedit. Predefined conversion functions are:-

ORD (x)	The ordinal value of the char (x). (its ascii code)
CHR (x)	The char whose code is (x).
CONV (x)	The Real corresponding to integer (x).
Trunc (x)	The Integer corresponding to the real (x).

The Lookup procedure locates a nominated file in the directory and if found returns a boolean True, and the Attributes of the file (see Prefix type Fileattr & Filekind).

Identify tells the system the program name. This is printed on the console before anything else. It is useful as a de-bugging aid, or to indicate the source of results when other programs are called from one running. It is printed once only for each phase. It is optional and is the only use of the pgm name during execution.

Run calls a program from one running. The name and params are exactly as would be typed for a console call, but two extra variables must be supplied as params. They are used after return to the caller, to show Where and How the called pgm was teminated.

Xpeek/Xpoke similar to Basic Peek/Poke.

Xcall calls an Assembler program into action from a running Pascal pgm. (I have not used this yet but it seems simple!

All routines starting IO....

A problem here as most use Prefix descriptions of peripherals such as Typedevice, Printdevice, etc. I have never had these accepted by the compiler which only recognises Mdex Cons.dev, Print.dev,etc. Such routines thus do not work. Exceptions are the types Ioresult and Ioparam. Both of these are effective.

Taskkind. Three tasks are defined, Inputtask, Jobtask & Outputtask. I have never been able to get out of Jobtask. Brinch Hansen says that Readline/Writeline are not effective in Jobtask. They don't work.

Program p(var param:arglist)

This is called when the pgm is activated from the console. No other definition Program must appear.

Other Features. Pre-defined operators for:-

Sets	OR	Union	= Equal
	&	Intersection	<> Not equal
	-	Difference	<= Contained in
	IN	Membership	>= Contains
	:=	Assignment	

Enumeration types
:= < = <= <> >=
These have the expected meaning. Results are always Boolean.

Boolean & OR NOT where results are either False or True.

Integer + - * DIV MOD

Real = < > <> <= >= + - * / the pre-defined function ABS(x).

Arrays := = <>

Strings < > <= >=

Records := = <>

Storage requirements in bytes
Enums:2; Reals:8; Sets:16; String of m chars:m;.

Control Chars. These must be written in ascii coded form. e.g. '(:10:)' is the form for Line Feed. They are string vars.

Some Standard Functions. For enumerations,
SUCC (x) and PREC (x) return the successor or predecessor value of x (if there is one).

Arithmetic Functions.
Brinch Hansen Ref[1] is silent on this topic. Some which are implemented are:-
ABS(x), SQR(x) square of x, SIN(X).
these return a result of the same type as x , but that must be Real or Integer.
COS(x), ARCTAN(x), LN(x) the natural log.
These must all be Reals.
There must be others, perhaps somebody knows?

Variable & Universal Parameters.
When a procedure is given parameters, it may normally use the value supplied but is not allowed to change it. These are Constant Params. If the procedure is required to change the value, the param identifier must be preceeded by the symbol VAR. e.g. Procedure Something (addr:integer; var block:page)
Of the two params the routine may change the value of 'block', but not 'addr'. Similarly, it cannot always be known in advance what type of data a variable will be called upon to accept. This may be accomodated by writing UNIV in front of the identifier.
e.g. Procedure Something (univ text:line).
This makes 'text' a universal param which can accept any data type occupying the same length as would type char.

Conclusion. Much more information can be got from reading print-outs of master disc programs of _____.pas.
Note that in an IF-THEN-ELSE statement, no semi-colon must be used in front of Else.
Is there anyone with original Mdex Pascal documentation prepared to make it available for copying?
Finally, Brinch Hansen mentions several console commands. I can't find any which work. Can you?

The Program Copy.

Call from console,
type 2/copy infile,outfile

Copy transfers data from in to out. The files can be disc files with Drive number, or device files, Cons.dev, Print.dev, etc. If you can't remember the correct call, just type Copy . A prompt message is printed.

```
#p60
111.
112.          {      End Prefix      }
113. (&)
114.      { Global Declarations for Copy.}
115.
116. Const eot = '(:04:)';  nul = '(:0:)';
117.    { These are String Constants but control chars }
118.    { must be written as decimal coded Ascii.}
119. Var s:integer;
120.      ok, console, found: boolean;
121.      attr:fileattr;
122.      text:line;
123.
124.      { Procedure declarations }
125.      { Note: If a procedure calls other procedures, its }
126.      { declaration must follow any for those it calls. }
127.
128. Procedure Context (text:line);
129.  Var i:integer;  c:char;
130.       {local variables only exist whilst the procedure is }
131.      { being executed. They are not accessible from outside.}
132. Begin
133.    i := 0;
134.    Repeat
135.      i := i+1;  c := text[i];
136.      Display (c);
137.    Until c = nl;
138. End;    { of procedure Context }
139.
140. Procedure Help;
141. Begin
142.    Context ('Bad call. Correct form is ''Copy (Infile,Outfile:identifier)''(:10:)');
143. End;
144.
145. Procedure Error (text:line);
146.       {the string passed to this procedure is passed on again to Context}
147. Begin
148.    Context (text);
149.    ok := false;
150. End;
151.
152. Procedure Set_files;  { loop to locate & open both files }
153. Var filename:argtype; i:integer;
154.      {Note this var i is not the same as that used in Context.}
155. Begin
156.    ok := true;
157.    for i := 2 to 3 do
158.     Begin
159.      with param[i] do
160.      if (tag <> idtype) then Help else
161.      begin
162.         Lookup (id, attr, found);
163.         if found then
164.       Begin
165.         Case attr.kind of
166.            scratch,concode,seqcode:
167.             Error ('File kind must be ascii(:10:)');
168.            Ascii:
169.         End;    { of Case }
170. (&)
```

```
#P*
170. (&)
171.              If i = 2 then Writearg(inp, param[2]) else
172.                            Writearg(out, param[3]);
173.
174.       End      (of If found...,NB no semi-colon here. )
175.     Else
176.        If i = 2 then Error('IP file unknown(:10:)')
177.                 else Error('OP file unknown(:10:)');
178.     End;    (of If tag...else)
179.    End;     (of loop)
180. End;        (of procedure Set_files)
181.
182. Procedure Terminate;  (Closes files & vets performance.)
183. Begin
184.     Write (eot); write (em);
185.     Readarg (inp,param[2]);  ( close IP  & return status.)
186.     if not param[2].bool then
187.        error('Problem reading IP file(:10:)');
188.     readarg (out, param[3]);  (return OP status )
189.     if not param[3].bool then
190.        error('OP file lost(:10:)');
191.     With param[1] do
192.     Begin tag := booltype; bool := ok end;
193.     ( use the extra param to set performance flag )
194. End;  ( of Terminate )
195.
196. Procedure Copytext;
197.     ( Copies standard pages of ascii text.)
198.     ( Declare local variables for use )
199.     ( within this routine only. )
200. Var block:page;  eof:boolean;
201. Begin
202.   Repeat
203.     Readpage (block, eof);
204.     Writepage (block, eof);
205.   Until eof;
206. End;   ( of Copytext )
207.
208.     ( end of all preliminary definitions & declarations )
209.
210.     ( The main program )
211. Begin
212.   Identify ('  Copy:(:10:)'); (pass pgm name to system.)
213.   Set_files;   ( call this procedure )
214.   copytext;    ( call another )
215.   Terminate;   ( and the last.)
216. End.          ( note the mandatory point here )
217.
218.     ( End of Copy )
*Eof*
*
```

Oliver Hulme Hednesford, Staffs.

If Prem Holdaway or anyone else is still having trouble with the control keys of "MISSILE COMMAND & CANYON" games. I think they will find that all the variable "A" values require to be increased by 12. A rectified listing follows.

CANYON

```
680 IF A=2303 THEN P1=-1:GOTO 350
690 IF A=2559 THEN P1=1:GOTO 350
700 IF A=21247 AND P0>80 THEN P1=-80:GOTO 350
710 IF A=18175 AND P0<800 THEN P1=40:GOTO 350
720 IF A<>17919 THEN GOTO 350
```

MISSILE COMMAND

```
1070 IF A=16895 OR A=21503 OR A=17663: GOSUB 1190
1080 IF A=2303 AND PX>7:PX=PX-4:GOTO 1120
1090 IF A=2559 AND PX<240: PX=PX+4:GOTO 1120
1100 IF A=3071 AND PY>8: PY=PY-4:GOTO 1120
1110 IF A=2815 AND PY<152:PY=PY+4
1220 IF A=21503: GOTO 1270
1230 IF A=17663; GOTO 1280
```

Line 1140 will not run, I substituted F730 with F7=3 I can't guarantee this is correct but at least it runs. Another point worth a mention is that lines 1330,1460 and 1820 are MULTI-WAY CONDITIONAL GOTO lines. They will not run if 'THEN' is abreviated to ':'

HAPPY SHOOTING

Into onto and out of E.Bus Part 4

Tim Gray

Adding extra 9902 serial ports

The Cortex manual tells us that adding extra input output ports to the Cortex is just a matter of adding hardware. This is not totally correct as even if the extra hardware were added the Monitor and system software would not know unless it were told.

The sytem software knows of the presence and location of input output drivers from a table starting at 0084. This table is 16 words long with one word entry for each unit. For 9902 devices the table contains the C.R.U. address of the 9902 device and for other units the table contains the start address of the device service routine. in order to differentiate 9902 base addresses have 1 added to them to make an odd number.

The table in the standard Cortex looks like this :-

```
0084 0808  start address for unit 1 screen driver
0086 0081  CRU base address for unit 2 0080 RS232
0088 0181  CRU base address for unit 3 0180 cassette
008A 0754  start address for unit 4 centronics driver
008C TO 00A2 all zero, unused units
```

So once a 9902 is added to the Cortex its CRU base address must be included in this table. ie to add a 9902 as unit 5 with a base address of 0240, location 008C should be set to 0241.
Using the basic statement BAUD 5,4800 will now initialise the unit 5 9902 to 4800 baud.

The problems dont stop here though because for some reason the Cortex has no interupt 4 connection to the E.Bus. As control of 9902 devices for normal input output depends on interupts something must be done to enable its use.

Interupt 1 patch

It is possible to use the interupt 1 connection on the E.Bus to handle 9902 by performing the following patch.

Change 0004 to ED22 to set int 1 WP to the correct value
Change ED3A to F200 set user vector to patch start address

Enter the patch programme:-

```
F200 F300             WP
F202 F204             PC
F204 020C LI R12,>0000
F208 1E02 SBZ  2          disable -INTEN
F20A 0420 BLWP @>0010     branch to interupt 4 service
F20E 1D02 SBO  2          enable -INTEN
F210 0380 RTWP
```

REMEMBER TO SEND IN YOUR ARTICLES FOR THE NEXT NEWSLETTER

This patch passes all level one interupts to the interupt 4 service routine. To use it setup the code as above, it need not be at this location, then set board rate to enable input or unit to enable output. Then enable interupts :- BASE 0: CRB(2)=1

The extra 9902 on the E.Bus should now respond as normal.

Hardware mod

As an alternative approach you may wish to permanently modify the computer for interupt 4 from the E.Bus in place of interupt 1. If you do here are the details :-

Cut the track linking IC16 pin 4 to pin 2
Connect IC16 pin 4 to pin 6
Cut track linking IC16 pin 3 to IC61 pin 5
connect IC16 pin 3 to IC16 pin 1

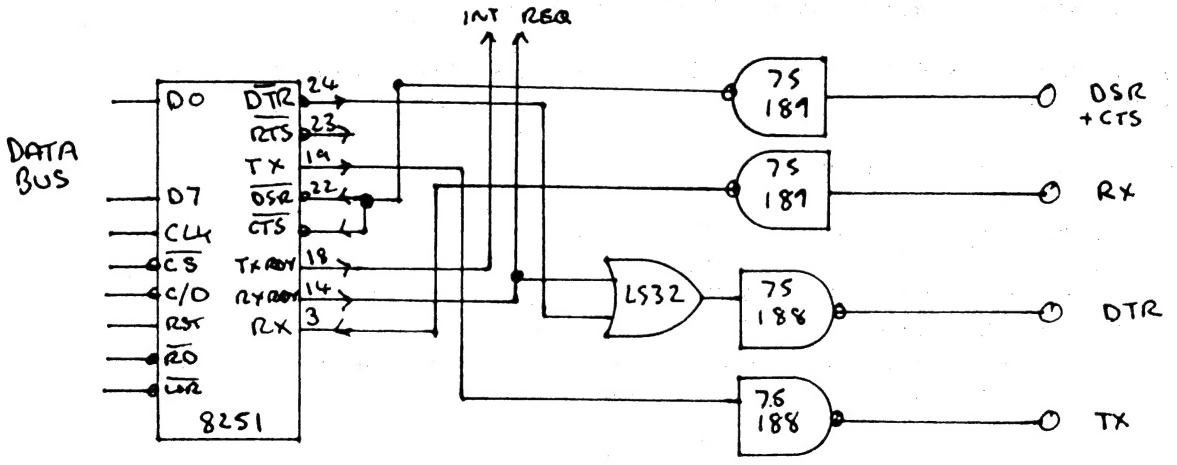
Bus -INTEN will now cause an interupt 4 request.

Extra 9902 circuit

I have presented here a Circuit of a 9902 E.Bus card that I have installed on my Cortex and tested in both of the above modes of operation.

Note that the -CTS input is connected to the -DSR input this allows faster responce to a request from the receiving device not to send more data yet. It is a simple mod that can also be performed on the internal RS232 9902. Also the circuit feeds an inverted -INT out as CTS this works if receive interupts are enabled as it stops the sending device from sending any more data until the interupt has been serviced.

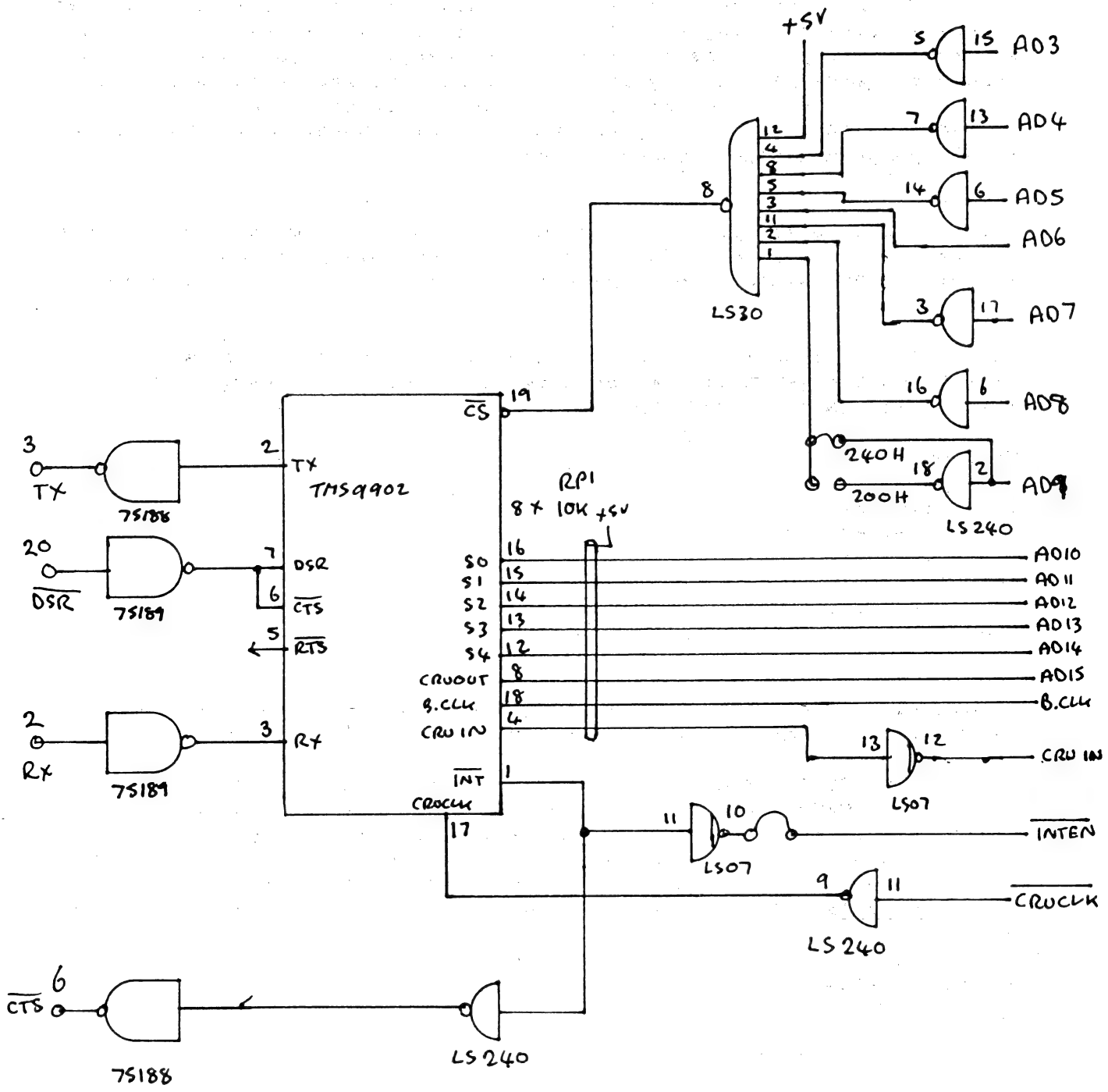
These simple changes to the normal circuits allow a much faster data rate to be used without feer of loss of data by using more efficient handshaking. A simmilar mod can also be used on Z80 type circuits using the Intel 8251 by utilising the RXRDY output that is usually used to request interupts.



REMEMBER TO SEND IN YOUR ARTICLES FOR THE NEXT NEWSLETTER

E.BUS 9902 SERIAL PORT

T. GRAY.



Port Addr 240H OR 200H

P.C.B.'S AVAILABLE FOR £1-00. FROM TIM GRAY.

A QBASIC Programme can be split into seperate sections & compiled seperately.The advantages are:-compilation is quicker as only a small section of the programme needs to be be compiled instead of the whole programme,handy if the programme has to be modified at a later date. Also a library of useful routines can be compiled, stored & used as needed in future programmes,thus saving time.A Qbasic programme may be a Main programme plus Sub programmes.

The Qbasic Compiler joins your programme plus programmes from its own libiary at compilation time to form an executable programme,to do this it uses a MDEX system programme LINK to join the various programmes.

On the Qbasic disc is a file called QLINK, it contains commands for LINK.To enable me to link my programmes in Qbasic I have added the instruction IN @2/COMMAND.LNK.

File QLINK

```
in baslib.rel
in @2/command.lnk
loc qbasic.loc
fetch
in backstop.rel
map
end
```

2/Command.lnk is a file on Drive 2,which I have created. When writing my Qbasic programmes , it would hold linker instructions such as

LOC 2/SALE.LOC 2/NUMBER , which identifies each of the file names (2/SALE,2/NUMBER) as a table of contents files. Also you can switch from one set of contents to another,simply by inserting a period in column one of the file.If a period is in column one then the Linker will take no notice of the instruction (see me for modified linker) In the example below the only instruction in force are for programme 2/SALE.

File 2/COMMAND.LNK

```
.               COMMAND.LNK   DISC-ND1
.               ___________   ________
.Sale
.____
loc 2/sale.loc 2/number.loc 2/y-n.loc 2/key.loc
.
.Heat
.____
.loc 2/heat.loc 2/number.loc 2/space.loc 2/uline.loc
.
.Zrm-Budget pgm
._____________
.loc 2/zrm.loc 2/key.loc 2/number.loc 2/pound.loc 2/qprint.loc
```

If nothing is put into Command.lnk,ie:- if a programme is being developed that required no sub programmes the Linker will output an error msg but will continue linking.The files suffixed .loc are the table of contents files,each line will identify the relocatable file (sub or main programme) & its external symbols.

File 2/NUMBER.LOC

```
2/number.rel        num1,num2,num3,num4,num5
2/number.rel        num6,num7
```

The sub programme Number has seven external symbols num1 to num7, more of this later.

If the sub programme Number is a standard routine used in many programmes then it's relocatable file Number.rel can be put on the Qbasic system disc & Number.loc added to Qbasic.loc . If the sub pgm Number is used in a Qbasic programme only the label num1 to num7 need to be defined in the programme everthing else is then automatic . The above is not as complicated as it sounds,but in doing it this way it needs to be done once only .Compiling & Linking are simpler . More details of the Linker can be found in the MDEX Manuel.

If when writing your Qbasic Source files you are using the MDEX editor Window It can be configured so that on exit from the file it will automatically compile the source file.

If the Main programme uses two Sub programmes then the instruction to link is QLINKER MAIN=MAIN,SUB1,SUB2.If more than 2 sub programmes are to be linked then the MDEX Linker would have to be used which is a lot more work,but if the Command.lnk & .loc files are set up then the single command QLINKER MAIN will link no matter how many sub pgms there are.If over a period of time the programme has to be modified all is automatic.

As stated earlier,routines may be compiled & saved for further use, if as I do you store them on a seperate disc,then you would save three files for each sub programme,the Source,relocatable & .loc file , so a routine called Number would have the following files.
NUMBER.BAS
NUMBER.REL
NUMBER.LOC

When compiling a Main programme then Number.rel & Number.loc need to be copied onto the same disc as Main at linking time . If it is a standard routine then the contents of Number.loc will already be in Qbasic.loc & Number.rel will be on the Qbasic disc , then after the labels have been defined in the Main programme Linking is automatic.

The above sub programme Number is an input routine it will accept numerical input only or alphanumeric input, the number of chars on input can be fixed & various input messages can be displayed . All the various types of input can be set up in the main programme then passed to the Sub programme.I'll start with the sub programme Number & show how it used in the Main programme.

First the Qbasic statement:-

```
ENTRY      external name=internal name
```

Therefore the Sub programme Number uses a variable LHOLD% (see below) the value of which is made available to the Main programme or other Sub programmes.NUM4 is the external name,LHOLD% is the internal name.

```
ENTRY    NUM4=LHOLD%
```

So Number makes available the value in LHOLD% via NUM4 this value can be changed or tested by any other programme.Note one external name must be reserved for the programme Defined name in this case it is FN.NUMBER.

```
EXTERNAL  external name=internal name
```

External is complementary to Entry it declares that the internal name is not defined in this programme but is defined in some other pgm which has the external name in an Entry statement,so for the Main pgm to use the variables in Number then they have to be declared in the Main programmes EXTERNAL statements.ie:-

```
{MAIN.BAS}

EXTERNAL    num1=fn.number,num4=lhold% {define variables etc in Number}

lhold%=4                                {lhold% to pass to number      }

DUMMEY=FN.NUMBER                        {call sub programme number     }
END
```

If a name is put in the Subprogram statement then the name will be printed on the Linker memory map.Note the External statement the pgm is using variables from the Qbasic library.The Qbasic varibles are few as the Manual does not give much information .The construction of Modules is non-existant I've had to sort it out myself , so feedback would be welcome.

After number has been called it will return to the calling programme similar to a subroutine call.Below is a programme segment from Number. Any reference to variables includes strings , subscripted strings & subscripted variables.

```
{STD PROGRAMME MODULE  N D Osmond  OCT 1986 .
--------------------  ----------  ---------

FILE NUMBER.BAS
---------------
                This routine will except numerical
                input only plus a decimal point
                the input length can be controlled.
                Or char depending on Char.flag%.
                Inputs.
                CHAR.FLAG%.set for char/reset for numbers.
                LHOLD%.....Input length.
                LABEL$.....Input prompt.
                Output.
                HOLD$......Output
                DEC%.......Set no decimal input.
                MINUS%.....Set no minus input.}

SUBPROGRAM NUMBER.n

ENTRY     num1=fn.number,num2=hold$,num3=label$,num4=lhold%,\
          num5=char.flag%,num6=dec%,num7=minus%

EXTERNAL cchech=cchech%


DEF FN.NUMBER
%DEBUG
CCHECH%=0                                    {no echo on input}
BSPACE$=CHR$(8)+" "+CHR$(8)+CHR$(7)
CLEAR$=" "*35

               {pgm body}
               ......
               ......
               ......

FEND

END
```

Strings may be used in the Entry & External statement so can Subscripted strings & variables in the form ENTRY main1=WEEKDAY$() note parenthesis is left blank.This is the last feature on QBASIC hope you have all found it interesting.

Cortex User Group Sale Items

Hardware

R.G.B. interface P.C.B	£8.00
Centronics P.C.B	£7.00
E.Bus 512K DRAM P.C.B plated through hole	£40.00
Externel Video interface P.C.B	£15.00
Disk controller WD2797 + P.C.B Cortex I	£55.00
Disk controller WD2797 + P.C.B Cortex II	£60.00
E.Bus interface complete Kit	£30.00
E.Bus 8 X 8K EPROM socket card built but no EPROMS	£30.00
E.Bus 4K RAM 8K EPROM socket 16 I/O lines ex equipment	£15.00
TMS9902 UART IC's	£2.00
74LS612 Mapper IC's	£25.00
74LS611 or 74LS613 Mapper IC's (req pull up R's)	£10.00

Other IC's in stock please write in for quote

Software all disk formats please specify when ordering

CDOS basic disk system 1.20 for TMS9909	£45.00
CDOS basic disk system 2.00 for WD2797	£45.00
Wortex word processor + spelling check by J.Makenzie	£15.00
Drawtech graphics drawing package by Tim Gray	£20.00
Menue generator by A.R.C.Badcock	£10.00
Two pass assembler by R.M.Lee	£14.00
Two pass asembler by C.J.Young	£15.00
Cdos utilites disk - copy charge only	£2.00

Cdos programes and games all £2.50 each :-

ARCHIE	ASTEROID	BREAKOUT	BURGLAR	CATERPIL	C-PEDE
CANYON	COTELLO	FIREBIRD	FROGGER	GDESIGN	GOLF
HUNCHBACK	INVADERS	MAZE	MAZE-3D	MBASE	MICROPED
MIS-COM	MUNCHER	NIBBLERS	N-ATTACK	OLYMPICS	P.BOAT
PENGO	PONTOON	RESCUE	S-ATTACK	SPACE-BU	THE-ZOO
TRAG	VADERS	WALL	X/0		

MDEX Software all formats please specify

MDEX CORE with debug monitor text editor and basic	£10.00
MDEX ASM & LINK assembler and linker	£10.00
MDEX SYSGEN system generation kit	£10.00
MDEX WORD word processor	£10.00
MDEX P.D.S. all the above in one package	£30.00
MDEX S.P.L. system programming language	£10.00
MDEX META compiler generator	£10.00
MDEX QBASIC basic compiler	£15.00
MDEX PASCAL sequential pascal	£10.00
MDEX WINDOW full screen editor	£15.00
MDEX SPELL spelling checker	£10.00
MDEX utilities copy charge only	£2.00

REMEMBER TO SEND IN YOUR ARTICLES FOR THE NEXT NEWSLETTER